IA *Battle for Wesnoth*

Projecte final de grau

Nom: Martín Dans

Tutor: Javier Béjar

Data: 16/10/2016

Resum

Aquest treball ha estat plantejat com un treball d'exploració personal on s'exploren tècniques d'intel·ligència artificial per a videojocs.

Concretament es desenvolupa una intel·ligència artificial capaç de jugar al joc *Battle for Wesnoth*. Aquest joc es presenta com un joc on la component d'estratègia és molt important i alhora conté molts elements que el fan ser un joc força complexa.

En el joc a més presenta la dificultat de gestionar diferents unitats intendents on la cooperativitat entre elles és molt important per a tenir succés en la partida, per tant s'han analitzat maneres de aconseguir aquesta cooperativitat.

S'estudien diversos mètodes i la seva adequació en el joc. Per exemple, algorismes de *pathfinding* i d'exploració de mapes, arquitectura *shout-ahead*, diferents heurístiques, identificació de grups en un mapa, entre d'altres.

Es proposa un algorisme que s'ha implementat per tal de veure la seva viabilitat. Aquests resultats s'obtenen de comparar la intel·ligència desenvolupada en el treball amb la intel·ligència artificial que ve implementada en el joc.

Abstract

This work was conceived as a personal exploration work which explores techniques of artificial intelligence for video games.

Specifically is developed an intelligent artificial intelligence that is able to play the game Battle for Wesnoth. The game is presented as a strategy game where the strategy component is very important, also the game contains many elements that make it quite complex.

It presents the difficulty of managing different units where cooperativity between them is very important to have success in it, so in the following document we have analysed ways to achieve this cooperativity.

We study various methods and their suitability in the game. For example, pathfinding algorithms and exploring maps, architecture shout-ahead, different heuristics, identification of groups on a map, among others.

It is proposed an algorithm that has been implemented in order to see its viability. These results are obtained by comparing the intelligence developed with the artificial intelligence that is implemented in the game.

Índex

1 Context

El treball consisteix en realitzar una intel·ligència artificial que sigui capaç de jugar al joc d'ordinador Battle for Wesnoth [1]. Ha de ser capaç de realitzar una estratègia sofisticada que tingui en compte tots els aspectes del joc per tan d'aconseguir la victòria.

El joc és OpenSource, per tant el codi és accessible per a tothom i no hi ha cap problema amb modificar-lo. El llenguatge de programació utilitzat és c++ i el desenvolupament del joc està plantejat com un projecte on qualsevol persona es pot afegir i col·laborar.

El projecte, per tant, no pretén innovaren l'àmbit de la intel·ligència artificial, sinó aplicar mètodes coneguts al joc Battle for Wesnoth, analitzar si s'ajusten i serveixen per a aquest joc en concret i aplicar un o més mètodes, amb les variacions que es necessitin, per tal d'aconseguir una intel·ligència capaç de jugar.

Per aconseguir aquest objectiu s'han d'analitzar i tenir compte els diferents aspectes del joc que condicionen l'estratègia a seguir.

- El caràcter del joc, ja que és un joc de múltiples peces per jugador on la cooperació de tot l'equip és essencial per la victòria.
- Els elements del mapa tenen molta importància, per tant hi ha un component d'anàlisi i identificació dels components d'aquest.
- Els diferents tipus de races presenten característiques diferents.

Gràcies a que el joc està pensat per ser desenvolupat per molta gent i que aquesta s'afegeixi en qualsevol moment del projecte, la documentació explica cada aspecte del codi i està preparada per no necessitar res més per poder entendre'l [2].

No només hi ha aquesta informació sinó que ofereixen una guia per a jugadors que permet aprendre més ràpid sobre el funcionament del joc on s'expliquen diferents estratègies [3].

El projecte va dirigit com a projecte personal i d'exploració. Es podrien beneficiar tots aquells jugadors del joc, ja que s'afegeix una nova intel·ligència al joc que fa servir una estratègia diferent a la ja implantada.

1.1 Estat de l'art

1.1.1 *Battle for wesnoth* dins del camp de la intel·ligència artificial

El camp de la intel·ligència artificial és un camp molt estudiat, i actualment s'estudien moltes branques i hi ha molts fronts oberts. Dins d'això, en aquest projecte s'enfoca a

una intel·ligència capaç de jugar a un joc, concretament un de simulació. Tot i així, seguim en un àmbit extens i estudiat, ja que molts dels jocs d'avui en dia tenen una component de intel·ligència artificial que és capaç de jugar i simular força bé el comportament d'un humà. Cada any es fan molts concursos on es proven les intel·ligència artificial amb diferents tipus de jocs [4].

Molts jocs però, per evitar perdre molt de temps amb l'execució de la intel·ligència artificial fan trampes i els jugadors controlats per la màquina tenen certs avantatges respecte els jugadors. Això és degut a que una bona intel·ligència artificial pot ser molt costosa en temps per tant, es dedica menys temps al càlcul de la intel·ligència artificial i se li permet tenir mes coneixement de l'estat del joc del que hauria per tal de compensar la dificultat i continuar sent un repte.

El joc inclou una intel·ligència artificial capaç de jugar, que és la proposada com a repte a guanyar als jugadors en el mode "campanya". A més hi ha altres treballs que han realitzat intel·ligències artificials pel joc, ja que, com s'ha comentat abans, és un bon joc per provar tècniques d'intel·ligència artificial.

Es pot trobar informació d'investigacions d'intel·ligència artificial d'universitats o de congressos d'alt renom [5] on s'expliquen diferents estratègies per desenvolupar intel·ligències per a videojocs.

1.1.2 *Pathfinding*

Battle for Wesnoth és un joc que es juga sobre un mapa dividit en cel·les, aquestes cel·les estan connectades entre elles de manera que els jugadors tinguin navegabilitat pel mapa. Clarament la definició d'aquest mapa la podem extrapolar a la de un graf, representació que farem servir pel mapa.

Per poder analitzar aquest mapa, necessitem un algorisme capaç de recórrer el graf buscant camins possibles, dels quals n'haurem d'escollir els més adients. Per fer-ho farem servir la família d'algorismes de *Pathfinding*, que recorren els nodes d'un graf, buscant camins entre un node inicial i un final.

Aquests mètodes exploren el graf de manera que, a partir d'un node inicial, visiten els seus nodes veïns, i així repetidament, fins que arriben a trobar el node final o no queden nodes per explorar.

Els algorismes més comuns són:

Cerca en amplada (BFS)

Aquest algorisme d'exploració de grafs, consisteix en anar visitant els nodes ordenadament per distància respecte l'origen. Per fer-ho, l'algorisme va augmentant el nivell de profunditat en cada iteració, de manera que per cada node s'exploren tots els

seus adjacents i aquests no s'exploren fins que la resta de nodes del nivell actual no han estat explorats.

És un algorisme molt útil si el que es vol es una exploració general del mapa, ja que et fa una exploració del mapa a distància k del origen i sempre ordenadament per profunditat, cosa que afavoreix a fer càlculs amb els camins ja que la distància acostuma a ser un factor clau. Com a punts contraris podríem veure que fer una exploració massa gran, pot repercutir molt amb el temps d'execució.

A*

Aquest és un dels algorismes més utilitzats per trobar camins entre dos nodes en un graf. Fa servir un heurístic per predir el cost que li queda per arribar al destí, de manera que pot orientar-se en la direcció, aparentment correcte, per trobar el camí adient. A més te una segona funció heurística que informa del cost real que portem acumulat per arribar a la solució parcial actual.

Si sumem el cost actual amb el predit, podem comparar els possibles camins i agafar el que tingui el cost més baix, d'aquesta manera podem evitar explorar molts camins innecessaris.

Com podem veure la força d'aquest algorisme radica en el fet de tallar molts camins possibles que no haurem d'analitzar inútilment, de manera que el temps d'execució serà inferior a una exploració completa del graf. El principal problema que presenta, és que dissenyar l'heurístic que prediu el cost pot ser una tasca difícil depenent del problema.

Un heurístic molt utilitzat i típic en *pathfinding* és la distància en línia recta que queda per arribar fins al final, ja que és un heurístic fàcil de programar i amb un cost baix.

1.1.3 Algorismes actuals d'intel·ligència artificial

A continuació es presenten algorismes destacats degut a la seva importància actualment i que s'han valorat alhora de desenvolupar la intel·ligència artificial.

Minmax

Si estem parlant d'un joc amb dos jugadors amb objectius totalment oposats com és el cas de *Battle for Wesnoth* llavors es podria fer servir un *minmax*. L'algorisme consisteix en trobar totes les possibles jugades fins a una certa profunditat i a partir d'aquí avaluar els estats i propagar-los de manera que s'alternen els estats mínim i màxim.

Per fer la propagació ens hem de fixar en el estat actual i els seus fills, de manera que si estem en un estat mínim, agafem el mínim valor dels seus fills i si estem en un màxim agafem el màxim. D'aquesta manera estem preveient que els dos jugadors executen la seva millor jugada. [9]

Montecarlo tree search

Es basa en trobar els moviments més prometedors, ampliant l'arbre de cerca amb un mostreig aleatori ponderat de l'espai de cerca. És a dir es fan moviments a l'atzar i si el resultat acaba en guanyat o perdut s'actualitzen tots els nodes de l'arbre amb el nou resultat. Les noves ponderacions d'aquests nodes depenen del resultat, de manera que els nodes amb millors resultats tenen més probabilitats de ser triats.

Es pot fer servir en jocs basats en torns on no hi ha informació amagada al jugador. Es una millora respecte a l'algorisme *minmax* ja que aquest col·lapsa amb espais de cerca massa grans. Per fer-lo servir no cal entendre el joc ja que no se li aplica cap tipus de coneixement a les jugades aleatòries que es trien [6].

Actualment s'ha fet servir per implementar una intel·ligència artificial del joc GO, joc molt conegut per la seva dificultat i que suposa un repte en el camp de la IA molt important.

shout-ahead

És una arquitectura d'intel·ligència artificial cooperativa que es basa en que cada unitat prengui individualment la seva decisió en una primera frase (*shout*) i després en una segona fase (*ahead*), sabent el que ha triat la resta d'unitats, es tria la decisió individual final.

Aquest mètode ha estat implementat i provat per investigadors de la universitat de Calgary, Canadà. Finalment van implementar una versió on s'utilitzava un aprenentatge híbrid que millorava els resultats [7].

1.2 Formulació del problema

L'objectiu final del projecte és haver aconseguit realitzar una intel·ligència artificial capaç de simular el comportament d'un jugador amb un nivell mitjà/expert en un joc d'estratègia determinat. Aquest joc és *Battle for Wesnoth* ja que compleix els requisits, és un joc d'estratègia amb una dificultat elevada, està ben documentat i es proporciona lliurament el codi.

Per tal d'assolir aquest objectiu es necessitarà:

- Analitzar la informació que proporciona el joc.
- Entendre el codi font del joc per tal de ser capaç de modificar-lo.
- Proposar diferents estratègies que es poden fer servir per obtenir la victòria i mètodes coneguts d' intel·ligència artificial que permeten recrear-les.
- Estudiar i valorar les diferents tècniques d'intel·ligència artificial rellevants per l'objectiu.
- Comparar i ajustar les estratègies per tal d'escollir la millor estratègia trobada.

- Implementar una versió final que sigui capaç de reproduir fins un cert punt el comportament d'un jugador.

Com que aquest projecte està plantejat com una evolució personal sorgeixen objectius secundaris personals que s'assoleixen en realitzar el treball i no com a be final d'aquest:

- Ser capaç d'entrar en contacte, entendre i modificar un projecte real.
- Entendre on està actualment la IA aplicada a jocs.

1.3 Abast

L'objectiu del projecte és aconseguir una bona intel·ligència artificial i valorar objectivament quin és el resultat obtingut, comparant-la amb la ja existent del joc. És centrarà sobretot en intentar imitar l'estratègia que faria un humà, per tant, els esforços aniran en aquesta direcció.

En el projecte és presenten les següents dificultats i és resumeix la idea de com es tenen previstes solucionar:

- La complexitat del joc. Un dels problemes de desenvolupar una estratègia és que per saber si es bona o no s'ha de tenir un coneixement profund del joc. El joc proposat és un joc que destaca per la seva complexitat. Per intentar evitar que les estratègies proposades siguin ineficients es demanarà ajuda si cal a un jugador expert i s'analitzaran conforme les recomanacions de tàctiques avançades que donen els propis desenvolupadors.

- Trobar el mètode que s'ajusti adequadament a la estratègia que es vol implementar. Com s'ha esmentat abans la quantitat de la informació sobre intel·ligència artificial és extensa, tot i així, cada problema s'ha de tractar per separat i saber valorar quin és el que permet tractar millor la informació de la que disposem és essencial així com la seva eficiència. Els problemes de videojocs han de donar una resposta en un temps ràpid, sinó les partides s'allargarien molt, per tant, és limitarà l'avant d'opcions a aquella branca de la intel·ligència artificial que estudiï especialment videojocs.

2 Planificació

2.1 Taula de tasques

Planificació proposada:

Tasca	Duració (setmanes)
GEP	6
Anàlisis del joc	2 (S)
Entendre el codi font del joc	1 (S)
Buscar mètodes	1
Definir l'algorisme	2
Implementació	4
Observar i comparar els resultats	1
Extreure conclusions	1
Preparació de la defensa del projecte	1
Correcció	3
Redactar el document	-
Total	19

(S) = Setmanes solapades amb una altre tasca, no es sumen al total.

- = Es realitza juntament amb la tasca realitzada.

2.2 Descripció de les tasques

A continuació es presenten les tasques del projecte amb el seu temps de realització estimat:

Aquest primer bloc de tasques són tasques que s'han de fer al principi ja que no tenen cap prerequisit, en conseqüència es poden fer simultàniament sense cap inconvenient.

- GEP (5 setmanes). Plantejament del projecte. Descripció formal, descripció de les tasques i costos, planificació temporal, etc.
- Anàlisi del joc (2 setmanes). Consisteix en un anàlisis en profunditat del joc identificant en aquest tots els elements importants i necessaris per a construir una bona intel·ligència artificial.
- Entendre el codi font del joc (2 setmana). Aquesta tasca consisteix en assolir els coneixements suficients per ser capaç de modificar el codi fet pels desenvolupadors del joc i poder implementar-hi la intel·ligència artificial.

A partir d'aquest moment les tasques tenen un ordre seqüencial i cada una es requisit de la següent.

- Buscar mètodes d'intel·ligència artificial que s'adeqüin a la informació que disposem del joc (1 setmana). Aquesta tasca és molt important ja que definirà les possibilitats del projecte. És buscarà una base sobre la qual treballar i poder desenvolupar la intel·ligència artificial.
- Definir l'algorisme que s'implementarà (2 setmanes). Es profunditzarà amb l'algorisme detallant cada apartat d'aquest.
- Implementació i integració de l'algorisme dins del joc (4 setmanes). En aquest apartat l'algorisme ja ha esta definit i s'implementa. Es contempla un temps de correcció dels errors que aquest pugui tenir, ja sigui degut al plantejament o a la implementació.

En aquest punt el que es podria considerar el cos del treball ja el tenim fet i el que falta és avaluar els resultats obtinguts.

- Observar i comprar els resultats obtinguts (1 setmana). En aquest projecte hi ha un apartat d'avaluació de la intel·ligència obtinguda. En jocs complexes no és fàcil trobar la manera d'avaluar si la intel·ligència artificial es bona o no, per això la manera de valorar-la serà comparar-la amb la que ja ve donada amb el joc.
- Extreure conclusions del projecte i explorar possibles ampliacions (1 setmana). Per últim s'extreuen les conclusions a les que es pot arribar després d'haver realitzat el projecte, com ha quedat la intel·ligència artificial finalment i si s'ha trobat alguna possible millora d'aquesta que es podria realitzar en un futur.
- Preparació de la defensa del projecte (1 setmana). Es preparà la presentació oral del projecte.

Al final de cada bloc és deixa una setmana per a revisar la feina amb el tutor i corregir puntualment allò que calgui millorar. Aquestes són petites correccions i retocs finals no grans canvis ja que les directius generals hauran estat marcades abans. Per tant afegirem la tasca:

- Correcció (3 setmanes).

Durant tot aquest temps es realitzarà una tasca continua que és la redacció del document final del projecte.

- Redactar el document (Sempre).

Per desenvolupar les tasques es disposen de 375h. El projecte té una durada de 18 setmanes, per tant es comptabilitzen 21 hores de feina per setmana.

2.3 Metodologia i rigor

És molt difícil avaluar la qualitat d'una intel·ligència artificial. Normalment alhora d'avaluar una intel·ligència artificial resolent un problema s'assumeix com a estàndard una determinada intel·ligència artificial que és acceptada per tothom com a bona i a partir d'aquesta es comparen la resta amb aquella.

D'aquesta manera es fa jugar a la intel·ligència artificial estàndard contra la que es vol provar per veure el rendiment que aquesta presenta. En aquest projecte es seguirà aquest esquema i s'avaluaran els resultats en funció del la IA que ve donada pel joc.

Durant el projecte s'anirà revisant la feina feta. En l'apartat de formulació del problema, es veu com s'ha dividit el treball en objectius.

Per cada objectiu es buscarà informació i es plantejarà amb el tutor. D'aquesta manera s'eviten errors de plantejament.

Posteriorment es durà a terme la recerca i programació d'aquell punt.

Finalment es farà una revisió amb el tutor que indicarà si s'ha realitzat correctament o no, i en cas que no ho sigui s'ajustarà a les recomanacions del tutor.

Després d'haver fet totes les modificacions al joc, aquest ha de seguir funcionant amb la nova intel·ligència artificial implementada. Llavors es podrà valorar la qualitat d'aquesta i el succés del treball en aquest sentit.

2.4 Valoració d'alternatives i pla d'acció.

Per a les tasques del primer bloc no hauria d'haver cap retràs. En el cas de de que es compliqués excessivament entendre el codi, el temps es trauria de la programació d'aquest. L'ideal en aquest punt és tenir una bona base i haver solucionat tots aquells grans dubtes que es poguessin tenir.

Els problemes que es presenten en el segon bloc no són un problema de temps sinó que el temps limita la profunditat a la que es pot estudiar i desenvolupar la intel·ligència. Per tant, si no s'assoleix l'objectiu aquest repercutirà en la qualitat del treball no en el termini d'aquest. Si no s'assoleix la qualitat mínima de la feina aquest es proposaria per a l'entrega del quadrimestre següent revisant adequadament la feina feta. Per tant, s'ha de tenir en compte que el treball es pot allargar en aquest aquesta fase tot i que no hauria de passar.

Per últim la feina l'últim bloc no presenta problemes, i es deixa el temps suficient per a acabar de tancar el treball.

Veiem que el punt crític del treball és la formalització i implementació de l'algorisme.

3 Pressupost

3.1 Identificació dels costos

En el meu projecte la majoria de tasques necessiten del mateix, per això trobo que no té molt de sentit dividir els costos per tasques a realitzar.

Per a dur a terme el projecte es necessita:

- Realitzador del projecte. Persona que desenvolupa el treball. Al ser un treball, aquesta persona fa tant l' anàlisis com la implementació de l'algorisme i realitza totes les tasques presentades anteriorment.
- Tutor del projecte. Persona que supervisa i guia el treball.
- Com a mínim un ordinador amb connexió a internet. Es consideren dos com a mínim contant l'ordinador del tutor. L'ordinador és l'eina principal de treball i el que suposarà el cost material més gran del projecte. Aquests ordinador no necessiten cap component especial tret dels més simples i que ja venen incorporats en qualsevol ordinador avui en dia.
- Font d'alimentació per als ordinadors.

Els possibles imprevistos del projecte es que s'allargués la durada, ja que descendir la qualitat del resultat final no provoca més cost, simplement menys rendibilitat. S'estima que amb 2 setmanes de més (un 25% del temps extra per al segon bloc d'activitats) es podria acabar encara que sorgís algun imprevist. En el pitjor dels casos s'estimen 4 setmanes. Si augmentem la durada del projecte sumariem cost de recursos humans i d'electricitat.

A més el projecte es desenvolupa en aquest entorn tancat i no hi ha cap causa externa que pogués augmentar els costos d'aquest.

3.2 Estimació dels costos.

Per començar en l'estimació dels costos mencionar que el projecte del TFG és una petita part de tot un projecte gran de desenvolupament d'un videojoc. Aquesta part ha estat feta per voluntaris, on invertien el seu temps personal sense cap retribució monetària. Per tant, en aquest apartat es tindrà en compte el cost del programador que teòricament inverteix unes hores en la programació i ho fa amb els seus propis recursos però en el projecte real aquests costs els assumeix la pròpia persona, que voluntàriament contribueix al projecte.

3.2.1 Estimació dels costos

El cost en recursos humans del projecte és:

- Programador: Són 375h de projecte. He suposat un sou de 1.5 cops el preu mínim que estableix la universitat. 375 * 1.5 * 7.5 = 4218.75 € .
- Tutor: He suposat que el tutor li dedica 2h a la setmana de feina i que el seu sou per aquesta feina seria d'uns 20 €/h. 2 * 18 * 20 = 720 €.

Per tant, en recursos humans es gastarien uns 4939 €

El cost en materials ve representat per:

- 2 ordinadors: En el cas que no es disposin d'ordinadors, per fer el projecte es necessitarien 2. He calculat a 500€ cadascun d'ells. 500 * 2 = 1000 € . Per una altre banda l'amortització dels ordinadors suposaria només de 50 *2 = 100 (Es suposa que s'utilitza un 10% de la seva vida útil).

Aquests costos són els únics que no depenen del temps del projecte ja que la vida d'un ordinador es molt superior a la durada d'aquest mateix i , per tant, amb una sola compra es més que suficient. Per tant estem amortitzant els ordinadors i es considera la part proporcional al cost de la seva vida útil utilitzada.

El cost en altres despeses és de:

- Connexió a internet: Si s'hagués de pagar la connexió a internet exclusivament per aquest projecte, s'haurien de considerar 5€ per setmana més. 18 * 5 = 90 €.
- Electricitat: Un ordinador al dia gasta uns 0.15 € al dia en electricitat. 0.15*18*7 = 18.9€. Sumant la electricitat de l'altre ordinador estimo 25 € en electricitat.

Estimació del cost del projecte sense tenir en compte possibles incidències.

4939 + 100 + 90 + 25 = 5154 €

Si mirem el projecte com un projecte de final de grau, en cas de que es donés que el projecte s'allargués un quadrimestre més el cost extra que s'afegiria projecte. No te sentit parlar de la probabilitat de que es doni aquest fet, ja que guardar una part proporcional dels diners necessaris per fer-ho no farà que es pugui realitzar. Com que és l'única solució que hi ha o es tenen els diners integres per allargar el projecte un quadrimestre sencer fins la nova data d'entrega o no té sentit guardar res.

Tot i això, el tems estimat d'allargar el projecte real són 2 setmanes. Com a molt es necessitarien 4 setmanes per repetir de nou tota la implementació. Com que calcular la probabilitat de que això passi és impossible ja que depèn de masses factors, s'ha sumat al pressupost 2 setmanes extres de feina.

Per això el pressupost real del projecte si el deslliguem de les entregues marcades per la facultat és de 5154 *2/18 = 5727.

El projecte té un pressupost total de 5727 €.

3.2.2 Taula de costos estimada

Font	Cost (€)
Programador	4219
Tutor	720
Ordinadors	100
Connexió a internet	90
Electricitat	25
Total	5154

3.3 Control de gestió

El control de gestió es dur a terme amb el tutor, a les tasques reservades per a fer-ho. Es decideix si la feina té la qualitat suficient o no i com es va mencionar en cas que no el tingui es rectificaria el que calgués. En cas que no es tingués temps per rectificar degut a que la feina té massa poca qualitat, llavors el projecte s'allargaria, com que ja es reserva per fer les modificacions oportunes només s'allargaria en el bloc 2 on es calcula que es pot solucionar qualsevol problema que hagi sorgit en 4 setmanes.

Això faria pujar el cost del projecte en 572 € (5154 * 2/18).

Tot i que a 2 de 3 blocs de tasques no es considera que hi pugui haver cap endarreriment i el temps assignat a cada tasca hauria de ser suficient, no es pot garantir que el projecte segur que s'acabarà en el temps establert ja que sempre poden sorgir errors durant la implementació de l'algorisme. Per això s'entén que és molt important partir d'una bona base d'anàlisis del joc i de recerca d'informació en les activitats del bloc 1.

4 Sostenibilitat

4.1 Matriu de sostenibilitat

Per últim s'inclou la matriu de sostenibilitat.

	PPP	Vida útil	Riscos	Total
Ambiental	9	18	-1	26
Econòmic	9	19	-1	27
Social	7	18	0	25
Total	25	55	-2	78

4.2 Dimensió econòmica

En aquest projecte existeix una avaluació dels costos.

El cost del projecte no és competitiu ja que aquest projecte s'està fent de manera altruista, i qualsevol cost ja el fa perdre davant la iniciativa real.

Per desenvolupar el projecte els recursos que es necessiten són els mínims però el fet de que es pugui realitzar amb menor cost és perquè els assumeix la pròpia persona individual no el col·lectiu del projecte.

En aquest projecte es dedica més temps a allò que es necessita sense oblidar el temps de plantejament per tenir una bona base per desenvolupar-lo.

4.3 Dimensió social

El joc pel qual es desenvolupa una intel·ligència artificial necessita d'aquesta i és una component molt important i imprescindible d'aquest. Però la necessitat no és real ja que aquesta ja ha estat desenvolupada i integrada en el joc.

El projecte no millora res del joc actual, és un projecte d'àmbit personal, per tant, no afecta en res als usuaris actuals del joc.

Al no afectar el projecte original del joc, no hi ha cap tipus de perjudicat.

4.4 Dimensió ambiental

El cost del projecte és bàsicament electricitat.

Un cop acabat el projecte ja no es consumirà res, entenent que la intel·ligència artificial no influeix en que el joc funcioni o no. Amb aquesta idea, el projecte acaba un cop realitzat finalitza completament.

Els recursos utilitzables són els ordinadors.

Els ordinadors consumeixen aproximadament 1 kWh al dia d'energia.

No es genera cap tipus de contaminació durant el projecte, ja que tot es programació. Com a molt es tindria en conta els residus que es generen al acabar la vida útil d'un ordinador.

El problema es la importació de les matèries primeres que hi ha actualment en el món des de països subdesenvolupats a països desenvolupats i alhora utilitzar aquests països menys desenvolupats com a vertader de residus. Tot i això no considero que el meu projecte afecti positivament ni negativament a aquest problema, però òbviament utilitzar material d'empreses que tracten adequadament els residus és vital per a l'impacte mediambiental.

5 *Battle for Wesnots*: Característiques i dificultats

5.1 Anàlisis del joc

El joc *Battle for Wesnoth* és un joc d'estratègia per torns cooperatiu. La intel·ligència artificial que es desenvoluparà ha de ser capaç de fer cooperar diverses unitats cada torn.

5.1.1 Informació bàsica

La informació bàsica de la que es disposa en el joc és la següent.

- Per cada unitat:
    - Vida
    - Nivell
    - Experiència
    - Dany
    - Tipus d'unitat
    - Bonificació terreny
    - Posició
    - Àrea d'influència
    - Quantitat de moviment
    - Cost
    - Per cada tipus de terreny
        - Probabilitat d'esquivar atac
        - Cost de moviment per travessar-lo
- Per cada cel·la:
    - Tipus de cel·la
    - Efecte

Objectiu principal:

- Acabar amb l'heroi enemic

1 Imatge on es veuen cases, diferents tipus de terrenys i unitats

5.1.2 Accions

Una unitat pot realitzar diverses accions en un torn.

Les accions que no acaben el torn són moure's i reclutar més unitats. L'acció de conquerir una casa no finalitza el torn però si que esgota tots els punts de moviment que tingui la unitat. Per últim, un cop una unitat a executat el seu atac ja no pot realitzar cap altre acció acabant així el seu torn.

Només els herois situats a una cel·la de castell poden reclutar noves unitats.

5.1.3 Qüestions

A continuació es presentaran els problemes que s'hauran de resoldre durant el desenvolupament del projecte:

- Mapa (trobar camins).
    - Aconseguir fronts per no deixar passar les tropes enemigues al teu territori.
    - Aconseguir el màxim de cases per obtenir més or.

- Combat.
    - Atacar i moure als personatges en l'ordre correcte. Pot variar molt l'estratègia i el resultat depenent de l'ordre de moviment.
    - Defensar les tropes amb menys vida. Després de moure vigilar que les tropes al davant siguin les mes fortes i protegeixin aquelles mes febles amb les seves zones de control.
    - Decidir si retirar una tropa i enviar-la a una casa curar-se o enviar-la per matar un enemic crític. Si es sacrifica una tropa ha de ser per obtenir un valor a canvi sinó la majoria de cops simplement es retira si es pot.
    - Atacar quan l'hora del dia i el terreny siguin favorables a les nostres unitats.
- Reclutament de tropes.
    - Reclutar les tropes necessàries en cada moment.

Aquestes són les decisions a les que s'haurà d'enfrontar la IA. En un determinat estat del joc es presenten totes aquestes decisions alhora, per tant, s'haurà de gestionar i ponderar la importància en cada moment. Amb aquesta informació podem tenir una idea clara de quins mètodes i tècniques d'intel·ligència artificial s'hauran de buscar.

5.2 Dificultats

La majoria de mètodes tradicionals per fer una intel·ligència artificial en *Battle for Wesnoth* no donen resultats bons, això dona a lloc innovar nous algorismes que siguin capaços de jugar a un nivell decent.

Primer s'ha de tenir en compte la flexibilitat del joc. El joc inclou una diversitat d'unitats i de mapes molt amplia que fan variar molt l'estratègia. No només això, sinó que com que el joc està pensat per ser construït entre tota una comunitat de desenvolupadors, permet que els usuaris creïn les seves pròpies campanyes, mapes o unitats amb esdeveniments propis. Això inclou intel·ligències artificials especifiques per la campanya o per una unitat en concret. És molt difícil identificar patrons de jugades o de posicionament d'unitats, ja que cada raça i mapa presenta característiques molt diferents entre si, inclús unitats de la mateixa raça tenen diferencies molt importants. Això arriba fins al punt de que dues unitats d'un mateix tipus amb qualitats [1] diferents prenen un rol diferent en la partida. Aquesta diversitat fa molt més difícil desenvolupar una intel·ligència artificial genèrica.

Un altre factor a tenir en compte és l'impacte de l'aleatorietat en els combats. El mal que es fa i rep en un combat depèn d'un nombre aleatori i, a més, aquest té un gran impacte en la partida. Per tant, l'estratègia pot variar completament després de cada moviment en funció del resultat del combat.

[1] En el joc existeixen qualitats que proporcionen bonus a les unitats.

Encara podria ser més imprevisible si tenim en compte que hi ha l'opció de jugar boira[2] i, per tant, podríem no tenir tota la informació de la partida i haver de predir el risc de trobar unitats enemigues al moure una unitat. L'algorisme desenvolupat en aquest treball tracta amb tota la informació, és a dir les partides no tenen boira.

5.3 Complexitat computacional

La complexitat computacional que presenta el joc complica encara més les coses.

Un mapa normal té 30x20 caselles amb 16 tipus de terreny. Cada casella que introduïm al mapa multiplica pel nombre de terrenys que hi poden haver. Això implica que les possibilitats són de 16^nombre de caselles. En un mapa normal hi ha 16^600 possibles mapes, sense tenir en compte les unitats que hi puguin haver i totes les possibles variables que comporta el fet de tenir una unitat col·locada en el mapa. Per tant els possibles escenaris que es poden donar representen un espai de cerca immens.

Si analitzem el factor de ramificació de possibles accions també trobem un espai de cerca molt gran. En molts jocs d'estratègia per torns, cada jugador mou una peça o fa un moviment relativament simple. Això es diferent en *Battle For Wesnoth*, un jugador mou totes les seves unitats i l'altre jugador respon movent totes les seves. Cada unitat te un ventall de moviments molt extens, és normal tenir mes de 20 possibles moviments per unitat i que un jugador controli una dotzena d'unitats. Afegint que el moviment d'atacar ramifica en varis possibles atacs i que aquests, com s'ha comentat abans, no tenen perquè encertar. Això fa que pensar o intentar predir molts torns per davant sigui molt difícil. I concloem que el factor de ramificació de l'arbre de possibles moviments en un torn és molt gran.

Observem doncs, que *Battle For Wesnoth* es un joc que presenta dos grans reptes en quan a complexitat.

5.4 Estratègia bàsica

Abans de parlar de l'estratègia bàsica s'ha d'entendre com funciona el joc. Com s'ha mencionat abans el joc té moltes variables, per això a continuació només s'explicaran els conceptes necessaris per a poder entendre l'estratègia proposada.

En *Battle for Wesnoth* cada unitat té la seva zona de control, cap unitat enemiga pot passar a través d'ella, a no ser que tingui una habilitat que li permeti. Per tant, unitats que estiguin agrupades formen un front on l'enemic no pot passar sense derrotar-les.

[2] Només veuerem les cel·les del mapa on tinguem unitats que es puguin moure. No es veuran totes les unitats enemigues.

Aquí és on entra la veritable interacció entre els dos jugadors en termes d'estratègia. Com a jugador vols arribar fins a l'heroi enemic, però alhora has d'evitar que les unitats enemigues arribin fins al teu heroi. Si l'enemic envia tropes per una zona que està descoberta molt probablement guanyi la partida, encara que no tingui l' avantatge en terme d'unitats, o cases.

Per això, controlar la major zona possible del mapa és molt important. Perquè això implica que controles les cases de la zona i cada casa produeix una quantitat d'or al torn i l'or permet reclutar noves unitats a la partida.

Per tant l'objectiu de l'estratègia es controlar més terreny que els contrincant, per així produir més unitats a la llarga i acabar sobreposant-nos sobre el contrincant.

L'estratègia proposada es basa en la afirmació de que per a que un exercit guanyi fàcilment a un altre en una guerra, aquest ha de ser tres cops més gran que el seu enemic. El que s'està dient amb la frase anterior és que no és proporcional la relació entre les unitats dels dos bàndols i les unitats que el bàndol guanyador haurà perdut.

Com que l'estratègia ha de permetre expandir-nos d'una bona manera, conservar tot el territori que ja tenim i alhora ha de defensar l'heroi dels atacs enemics el que s'ha fet és expandir les nostres unitats de manera que la majoria faci pressió en una zona del mapa i la resta s'enviïn les unitats que es necessiten per defensar. Òbviament això implica eliminar les unitats enemigues de la manera més efectiva possible.

Es proposa definir diferents rutes que s'expandeixen per el mapa quan estan a prop de la base aliada i es concentren de nou quan s'apropen a l'enemiga. Aquests camins són els que seguiran les tropes aliades i optimitzen la expansió tenint en compte el cost de moviment de les unitats que es poden reclutar.

Per últim comentar que degut a la diversitat que té el joc, hi ha moltes estratègies i que una possible millora seria considerar adaptar l'estratègia escollida en funció de l'estratègia del rival. Això donaria a partides molt més dinàmiques i molt més complexes.

5.5 Intel·ligència artificial per defecte

En aquest apartat s'explicarà com està feta la intel·ligència artificial que hi ha implementada actualment en el joc per després poder comparar-la amb la implementada en aquest treball.

La intel·ligència per defecte que ve al joc l'anomenen *release candidate action.* Es basa en tenir una llista de possibles accions anomenades accions candidates que són avaluades per a cada moviment i executades en ordre segons la puntuació.

De base es tenen les següents (entre parèntesis els punts que té l'acció definida):

Anar a (200.000): Moure unitats a les coordenades següents.

Reclutament (180.000): Reclutament d'una unitat.

Moure líder (140.000): Moure el líder a un dels seus objectius.

Moure líder a un castell (120.000): Moure el líder al castell més proper disponible.

Combat (100.000): Atacar les unitats que estiguin en el rang.

Curar-se (80.000): Moure unitats a les posicions on hi hagi cures.

Cases (60.000): Capturar una casa desocupada o del enemic.

Retirar-se (40.000): Avaluar si una unitat està en perill o superada per molts enemics, i si ho està retirar-la.

Amenaces (20.000): Avalua possibles amenaces futures que puguin venir. Té en compte coses com el valor de les unitats, com de fàcil es arribar fins elles, l'exposició als atacs. Cal tenir en compte que aquestes accions són simplement apropar-se als possibles enemics, normalment s'ha executat una de les anteriors accions ha realitzar, com per exemple, lluitar o curar-se.

S'ha d'entendre que les accions s'executen de dalt a baix i que si pots atacar mai conqueriràs una casa, el que està fent es definir un ordre de prioritats i en el moment que una acció compleix una determinada condició aquesta acció s'executa independentment de les accions que hi pugui haver al darrere.

A més la intel·ligència artificial té definits uns aspectes que serveixen per configurar el comportament d'aquesta. Aquests el que fan es modificar les condicions de compliment de les accions, per exemple, canvien la manera de seleccionar les possibles amenaces.

6 Algorisme

6.1 Plantejament a priori de l'algorisme

Per a desenvolupar l'algorisme s'ha dividit en subproblemes que en conjunt solucionen el problema de guanyar la partida.

La idea que es planteja és defineix a continuació. Un primer anàlisi on s'analitza el mapa del joc. S'executa al principi de la partida i proporcionarà diverses eines que serviran per guiar. Després hi ha un algorisme que s'encarrega d'analitzar l'estat global del mapa i defineix una estratègia global a realitzar que defineix instruccions a les unitats corresponents. Aquesta estratègia global es va revisant i adaptant segons les circumstàncies, tant per moviments inesperats del contrincant o perquè no s'esperaven els resultats obtinguts per culpa del component aleatori[3] del joc.

Tenim doncs un primer anàlisis del mapa on es buscaran les millors rutes per arribar fins la base del contrincant. Es defineixen els possibles camins tenint en compte els millors llocs per lluitar i les millors rutes per conquistar el màxim de cases. Amb això definirem certs fronts importants on les nostres unitats han de tenir el control.

Aquí també trobem el que s'ha anomenat mapes de control. S'ha de diferenciar aquelles zones del mapa on les nostres tropes es poden moure i moure ràpid i aquelles zones on les nostres unitats tenen avantatge respecte les unitats dels contrincants en combat. Per tant, es crearan dos mapes amb pesos que representaran la bonança dels aspectes anteriors.

A més aquest algorisme introdueix certs objectius secundaris a les unitats per tal de segueixin certes normes alhora d'executar les instruccions.

Un cop tenim l'anàlisi del mapa fet, executem cada torn la resta de l'algorisme que decidirà els moviments a fer.

Primer de tot, es divideix l'estat de la partida en diversos batallons. Entem un batalló com a conjunt d'unitats que es troben relativament a prop entre elles. La diferencia entre els batallons aliats i enemics és que només els batallons aliats reben instruccions d'estratègia.

També s'identifiquen quins d'aquells batallons estan en zona de combat. Aquesta zona representa el combat entre dos batallons, un aliat i un enemic. És simplifica el problema de guanyar la partida a guanyar els combats entre batallons.

Hi ha un sistema basat en regles que decideix en quin estat de la partida estem. Decideix quina és la millor estratègia a seguir globalment i dona una instrucció especifica a cada batalló. En aquest apartat s'avalua si les nostres tropes són superiors

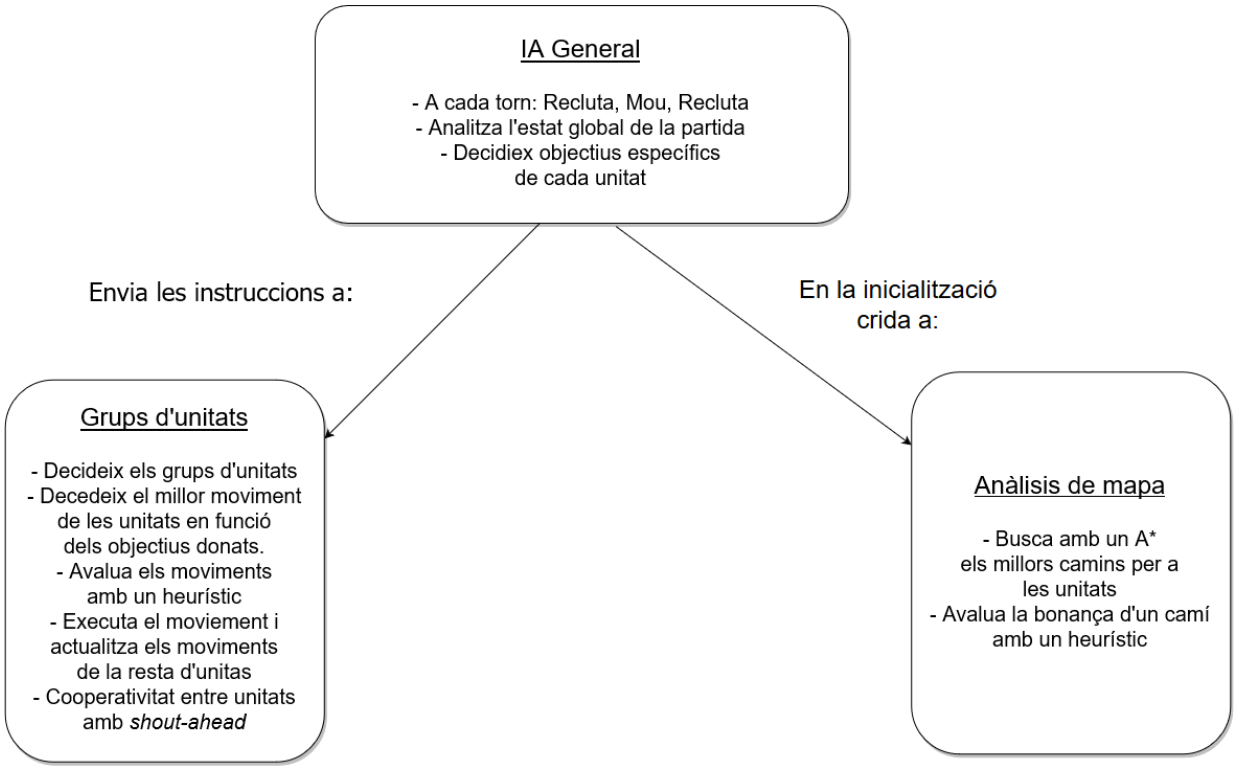
[3] Els càlculs que es fan es basen en el resultat més probable, però alguns rangs de possibilitats s'allunyen molt en els casos extrems i l'estratègia queda invalidada.

o inferiors a les de l'enemic, si mantenint els fronts es suficient per treure avantatge o si en canvi s'ha de fer més pressió. Podria ser que es decidís fer una jugada arriscada perquè estem amb molt desavantatge. D'aquesta manera cada batalló actua com a unitat independent en l'estratègia.

Un batalló que no està en combat segueix el camí assignat marcat pel mapa de camins que s'ha calculat en la inicialització. Mentre es segueix aquest camí s'envia a les unitats a conquistar les cases o cobreixen un front del camí segons s'hagi decidit.

Si hi ha unitats enemigues a prop és buscarà la millor manera de derrotar el batalló enemic fent ús de les unitats que es troben en el batalló aliat.

6.2 Estructura



2 Estructura i característiques principals.

Execució

Anàlisis de mapa

Jugar torn

Actualitzar grups d'unitats

Anàlisis de la situació

Analitzar possibles moviments

Per cada unitat aliada

Executar el millor moviment

Actualitzar possibles moviments

6.3 Definició i implementació dels subproblemes definits

6.3.1 Anàlisi del mapa

En la inicialització de la intel·ligència artificial es fa un anàlisis de mapa que crea els camins que es faran servir de guia durant la partida. Aquests camins serveixen per guiar les tropes des de la nostre base fins la enemiga. A més tenen la propietat d'anar pels terrenys més favorables de moviment per les nostres unitats.

Per tant, dividim l'anàlisi del mapa en dues seccions.

Primer analitzem quins tipus de terrenys són els que les nostres unitats poden passar fàcilment. S'assigna a cada tipus de terreny un pes, aquest pes es la mitjana del cost dels tipus de tropes que podem reclutar.

Executem un A* amb els pesos anteriors que troba el camí òptim per arribar fins la base enemiga. Aquest és el camí que considerem central, i és el camí més important.

Per trobar la resta de camins es modifiquen els pesos del mapa augmentant aquelles zones que es volen evitar. Es volen trobar camins que rodegin el camí principal per aconseguir-ho s'ha fet el següent:

- Calcular les celes que pertanyen a una circumferència amb diàmetre igual al 75% del mapa. A mesura que ens allunyem de la circumferència descrita s'augmenta el pes de la cel·la.
- Calcular l'àrea orientada que forma el triangle cel·la, base enemiga, base aliada. Segons els signe de l'àrea es suma molt de pes a les celes, per tal de forçar que vagi per l'altre banda del mapa descrita entre les dues bases.

Amb això obtenim diversos camins per la banda desitjada del mapa, que intenten obrir-se cada cop més fins que arriba un punt on comencen a convergir cap a l'altre extrem del mapa.

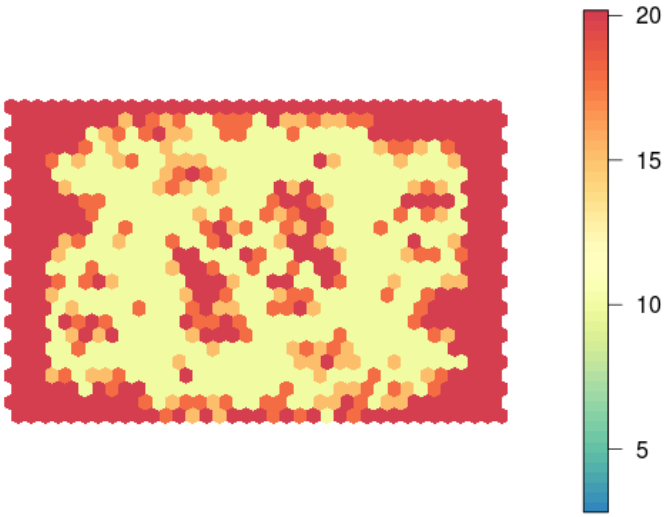
L'heurístic fet servir és la distància en línia recta que falta per arribar al destí i cap casella té un cost inferior a 1 per tant el resultat és òptim.

A continuació es té un exemple dels camins que s'obtindrien:

Veiem doncs perquè s'han triat aquests camins. Tots els camins compleixen els propietats que es busquen. Volem trobar el camí més ràpid, i alhora tenir camins auxiliars que ens permetin controlar el mapa i conquerir les cases.

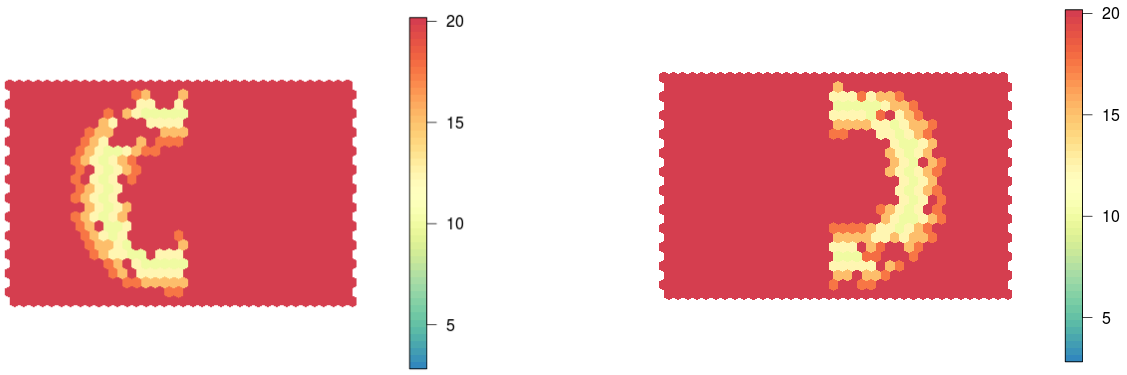
Tot i que aquest són els mapes més típics de *Battle for Wesnoth* hi poden haver d'altres en que si es fessin més camins milloraríem es tindria una millor aproximació del mapa, sobretot en mapes molt grans. La intel·ligència artificial implementada treballa amb tres camins perquè, en general, funciona bé, però una possible ampliació que milloraria l'algorisme seria determinar el nombre de camins a buscar.

En el mapa següent es pot veure quin és el cost que retorna la funció heurística per a cada cel·la del mapa típic:



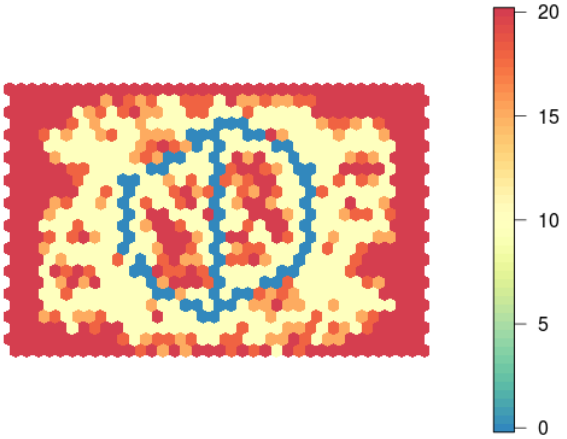
3 Mapa de costos

Ara veiem com es modifiquen aquests costos per tal d'aconseguir que els camins s'expandeixin cap als costats:



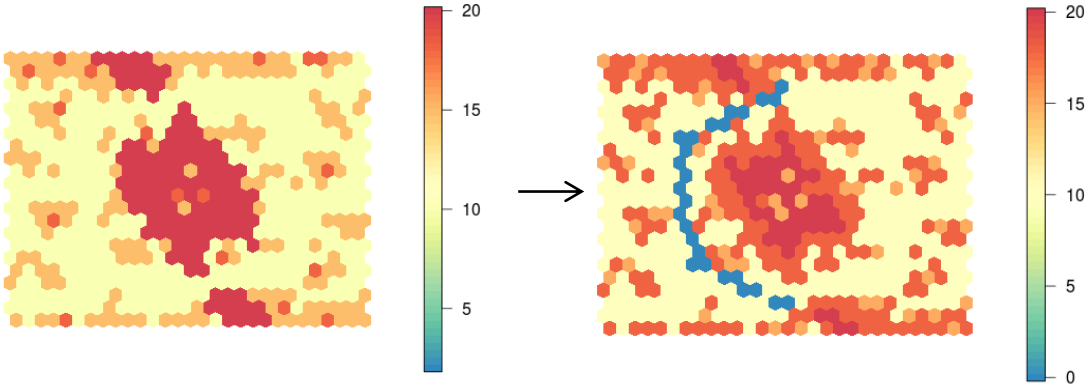
4 Camins cap als costats contraris

El resultat dels camins en el mapa anterior seria el següent:



5 Mapa amb els camins en blau

En un mapa on el centre no deixa pas el camí central s'adaptaria[4]:



6.3.2 Identificació de batallons

La identificació de batallons és molt important en l'algorisme desenvolupat ja que l'anàlisi que es fa per saber l'estat de la partida.

Per trobar els batallons s'executa un algorisme jeràrquic de classificació. A cada iteració del bucle s'agafen les dues unitats més properes i es mira si pertanyen al mateix al mateix grup.

[4] La raça amb la que s'ha calcula és diferent i com que es calcula el valor en funció de la raça que es té, la bora de la taca central és diferent.

Es considera que dues unitats pertanyen al mateix grup la condició que la distància ha de ser inferior a un màxim, sinó ho és, es considera que les unitats pertanyen a diferents batallons. A la inicialització de l'execució s'assigna a totes les unitats a un grup. Els dos grups passen a ser un sol assignant a totes les unitats al grup de l'altre unitat en el cas que es compleixi la condició.

El que s'aconsegueix amb això és agrupar aquelles unitats que estan prou juntes en batallons i separar els batallons quan es supera aquesta distància màxima. La condició d'aturada és per tant o bé totes les unitats pertanyen al mateix grup o les pròximes dues unitats més properes sobrepassen la distància màxima assignada.

6.3.3 Decidir instruccions

Representa la intel·ligència artificial global. Analitza l'estat dels batallons i decideix quines són les instruccions a enviar. Es decideix si estem expandint les tropes o estem retirant-les. De manera que cada unitat sap si ha d'intentar guanyar al batalló que té davant o millor retirar-se per conservar el màxim de valor possible. També indica objectius secundaris a les unitats per tal d'afavorir que es compleixin certes normes preestablertes que afavoreixen el comportament de les unitats i la cooperació entre elles.

En aquest punt s'han introduït diversos conceptes:

- Projecció de poder: Capacitat d'un jugador d'atacar a una cel·la en concret. És a dir, si totes les unitats d'aquest jugador ataquessin a aquella cel·la quin seria el mal estimat que farien. Aquest concepte és molt útil per saber si una unitat s'està exposant massa als atacs enemics.
- Percentatge avançat del camí: Donada una cel·la quin percentatge del camí que està seguint ha avançat. En aquest concepte és el que es basa la estratègia general, ja que les unitats molt avançades es comporten diferent a les acabades de reclutar.

Per a decidir els objectius secundaris d'una unitat es mira si es compleixen certes condicions i si ho fan llavors s'afegeix la instrucció secundaria corresponent a la unitat. La idea es que estem donant prioritat a una unitat a realitzar una acció en concret entre unitats que poden realitzar aquesta mateixa acció.

Es defineixen les següents:

- Si un camí té cases per conquerir, s'envien exploradors.
- Si no hi ha suficients exploradors s'envien unitats amb molt de moviment.
- Si un camí està a punt de perdre i necessita reforços s'envien unitats en aquell camí.
- Quan s'han enviat suficients reforços s'envien unitats pel nostre millor camí per seguir fent pressió al contrincant.

6.3.4 Moure individualment les unitats

En aquesta part trobem el que és el motor de la intel·ligència artificial. S'encarrega de moure les unitats seguint les instruccions que li arriben per a cada unitat de la fase anterior.

El cos principal és un bucle que s'executa fins que totes les unitats hagin realitzat el seu moviment. A cada iteració del bucle s'executa el moviment d'una unitat.

Inicialització:

- Agafar els possibles moviments de les unitats aliades.
- Per a cada possible moviment assignar un valor de bonança. Aquest valor és el resultat d'un heurístic definit al següent apartat.

A cada iteració es segueixen els passos següents:

- S'escull el millor moviment a realitzar entre totes les unitats i s'executa.
- Es tornen a calcular els moviments de les unitats i s'actualitzen els valors de bonança. Només es tornen a calcular aquells valors que han pogut modificar-se entre iteracions la resta s'agafa d'una cache per tal de millorar el rendiment.

6.3.5 Heurístic

L'heurístic és l'encarregat d'avaluar la bonança d'un determinat moviment, és a dir, calcular la avantatge que genera respecte el rival executar aquell moviment.

L'heurístic rep un moviment d'una unitat i les seves instruccions. S'executa una funció que retorna una ponderació per a cada component que es fa servir en l'heurístic en funció de la instrucció general enviada a l'unitat. D'aquesta manera s'adapta el comportament de la unitat seguint la instrucció general enviada per a la unitat. Per exemple, si estem atacant no s'avalua tornar enrere en el camí seguit i si avançar però si estem retrocedint les ponderacions s'inverteixen. De la mateixa manera que un atac retorna un valor més alt si estem atacant que si estem retrocedint. Totes les variables que avalua l'heurístic estan ponderades segons la instrucció general. Alhora suma punts extres si aquell moviment realitza un objectiu secundari dels que la unitat té assignats.

L'heurístic avalua punts per fer un bon atac, és a dir, aconseguir fer més mal en mitjana del que es rebrà a una unitat, té en conte si la casella final és una casa que donarà or i curarà la unitat, calcula si necessito avançar i assigna punts a les caselles

corresponents, calcula a la exposició que està la unitat i assigna punts segons l'ordre de prioritat que es vulgui moure la unitat.

L'heurístic fa servir una idea similar a la que es fa servir en l'algorisme de *shout-ahead* ja que calcula l'ordre que té aquesta unitat en fer una determinada acció. Així doncs una unitat guanya punts també si ha de tirar abans que l'anterior. Es va implementar una versió on es feia servir la *shout-ahead* arquitectura. Amb aquesta es tenia una predicció d'on volen anar la resta d'unitats i si aquestes tenen intenció de complementar al moviment actual. El moviment de la resta d'unitats es ponderava per simular la probabilitat que hi ha en que aquesta unitat faci un moviment que ens ajuda però va implicar un empitjorament en els resultats degut a la complexitat que suposava i finalment es va descartar de l'heurístic.

Els punts més importants que pondera l'heurístic són:

- Possibles atacs: Si la casella que estem avaluant permet executar un atac aquesta guanya o perd una puntuació en funció si l'atac es bo o dolent. L'atac és l'element més complexa a avaluar. S'ha considerat que un bon atac és aquell permet realitzar més mal del que es rep, deixa a l'enemic amb poca vida per a que un aliat el remati o directament l'atac té moltes possibilitats de matar l'enemic. Així doncs eliminar a un enemic dona molta puntuació ja que eliminar a una unitat enemiga proporciona molta avantatge.

- Conté una casa: Si la casella conté una casa se li afegeix molt valor també al moviment. Com ja s'ha vist abans les cases són un element molt important pel joc i especialment per la estratègia escollida. Depenen si la casa és aliada, neutral o enemiga s'assigna una puntuació diferent. Conquerir una casa ha de tenir molt valor, però no el suficient per evitar que s'eliminin les unitats enemigues quan toca. Per tant, dona menys punts que eliminar una unitat, però més que un atac mitjà o baix.

- Avançar en el camí: Un altre punt molt important és si la casella que estem avaluant ens fa avançar en el camí. Aquest component és el que fa avançar les unitats pel camí que tenen associat. En funció de la distancia a la que avança la unitat respecte la posició on està s'afegeixen punts al moviment.

- *Sanity check*: És una comprovació que es fa al final de l'heurístic on es comprova que el moviment sigui tingui realment algun valor. D'aquesta manera si estem exposant a molts la unitat a molts atacs enemics, no conquerim cap casa, ni realitzem un bon atac, aquest moviment no es bo.

Hi ha altres variables que s'avaluen però són casos molt específics, com per exemple si l'atac enverina a l'enemic o si la qualitat del terreny és molt dolenta. No s'expliquen amb detall ja que els punts que sumen o resten molts cops són irrellevants.

6.3.6 *Shout-ahead*, que va passar?

En la versió final del programa no es fa servir aquesta arquitectura. Tot i que es va implementar una versió on hi havia una primera fase on cada unitat decidia on volia anar i una segona fase on es decidia si es canviava l'acció en funció de la resta d'unitats, es va descartar perquè es va observar que al voltant del 80% de les vegades l'acció no es canviava i la majoria dels que es feia implicava un canvi poc significatiu positivament o un de molt negatiu. Això va ser degut a que els casos eren massa concrets i no es va saber generalitzar quan realment valia la pena adaptar-se a l'estratègia aliada abans de que aquesta l'efectués. Tenint en compte que a la iteració següent igualment s'actualitzaran els punts es va optar per eliminar la funció i deixar la cooperativitat en la pròpia heurística.

Això no vol dir que l'arquitectura no sigui viable, simplement que ajustar-la requeria molt més temps del que es podia invertir i es va estimar que els benèfics d'ajustar-la tenint una manera alternativa viable i el temps que calia per millorar-la no la feien eficient.

6.4 Complexitat temporal de l'algorisme

Analitzem les components per separat i després mirem la complexitat total l'algorisme. Es descriu que s'està calculant i el cost que té.

6.4.1 Anàlisi del mapa

- Iterar per tots els terrenys del mapa i calcular la bonança d'aquests en funció de les unitats a reclutar : Possibles unitats a reclutar * tipus diferents de terrenys.
- Buscar el líder entre les unitats. Com que només hi ha els dos líders en el mapa en aquest punt de la partida el cost és constant : 1
- Calcular els possibles camins:
    - Heurístic A*
        - Calcular l'àrea orientada : 1
        - Comprovar el tipus de terreny : 1
    - Executar l' A* per trobar els camins. El cost computacional de l' A* depèn molt de l'heurístic. En el millor dels casos (on l'heurístic és perfecte) l' A* té un cost computacional lineal mentre que en el pitjor dels casos no millora en res els algorismes de cerca típics i, per

tant, té un cost exponencial. Amb bons heurístics el cost tendeix a ser polinòmic. Per determinar el cost de l' A* s'han fet varies execucions i s'ha vist que el nombre de nodes explorat no és excessivament gran.

- Execució: Assumeixo que el cost és polinòmic i depèn de la distància entre líders. Experimentalment el nombre de nodes oberts mai ha superat distància entre líders³.

Cost d'anàlisis de mapa: Possibles unitats a reclutar * tipus diferents de terrenys + 1 + cost A* (distància entre líders) -> Normalment Possibles unitats a reclutar * tipus diferents de terrenys < cost A* (distància entre líder) -> cost A* (distància entre líders).

6.4.2 Identificació de batallons

- Mirar quines unitats estan més a prop: unitats²
  - En el pitjor cas estem traient una sola unitat per iteració: nº unitats iteracions
- Agafar la força: nº unitats

Cost d'identificació de batallons: unitats²* unitats + unitats -> nº unitats³.

6.4.3 Decidir instruccions (planejar estratègia)

- Identificar unitats perilloses. Es miren sempre les caselles de com a màxim radi dos del líder: 18
- Identificar castell líder: nº possibles moviments líder
  - Calcular possibles moviments unitats. Calcular els possibles moviments de les unitats implica mirar l'àrea d'una circumferència que depèn de la quantitat de moviment de la unitat. No és exacte però creixen igual en el límit: quantitat moviment² * nº unitats. Com que només volem una unitat: quantitat moviment líder²
- Identificar camins:
  - Calcular camí actual: longitud dels camins*nº unitats
  - Mirar si hi ha un altre millor. Com es que es comproven totes les unitats el cost és: nº unitats.

Cost de decidir instruccions: 18 + nº possibles moviments líder + quantitat moviment líder + longitud dels camins*nº unitats -> nº possibles moviments líder + longitud dels camins*nº unitats.

6.4.4 Moure individualment les unitats

- Escollir el millor moviment. Per escollir el millor moviment es té en conte els possibles atacs i on està la unitat situada en el camí actual: possibles moviments*possibles atacs + longitud camins
    - Calcular els possibles moviments (calculat a l'apartat anterior): quantitat moviment² * nº unitats.
    - A cada possible moviment s'executa l'heurístic. Aquest realitza moltes comprovacions però totes són en temps constant: 1.
- Actualitzar possibles moviment i actualitzar el valor de l'heurístic. Després de moure una unitat tornem a actualitzar els possibles moviments: escollir millor moviment * nº unitats

El cost total de moure les unitats és de: millor moviment * nº unitats -> quantitat moviment² * nº unitats².

6.4.5 Total

El cost total de l'algorisme desenvolupat és el següent:

Funció	Cost
Anàlisi del mapa	Cost A* (distància entre líders)
Identificació de batallons	nº unitats³
Decidir instruccions (planejar estratègia)	nº possibles moviments líder + longitud dels camins*nº unitats
Moure individualment les unitats	quantitat moviment² * nº unitats²

Cota superior de tot l'algorisme:

Cost A*(distància entre líders) + nº torns * (nº unitats + nº possibles moviments líder + longitud dels camins*nº unitats + quantitat moviment² * nº unitats²).

El cost més important i representatiu de tot l'algorisme és nº torns * quantitat moviment² * nº unitats².

6.5 Modificacions realitzades respecte el plantejament

La primera cosa que es va canviar respecte el plantejament a priori que s'ha comentat abans ha estat l'algorisme de moure les unitats. Per a resoldre aquest problema es volia fer servir en un primer moment un *montecarlo tree search* per fer una cerca d'estats. Es va valorar que l'algorisme podria funcionar però en fer un estudi sobre la ramificació de l'algorisme es va veure que el nivell de ramificació era excessiu inclús per a aquest algorisme.

El problema amb *battle for wesnoth* és que l'efecte aleatori del joc provocava no poder triar la següent jugada amb precisió i per conseqüència costa molt que l'algorisme arribi a convergir a una bona solució. Per conseqüència hi ha molt de soroll en les prediccions de si una jugada és bona o no ho és, pot ser que una jugada que ha resultat guanyadora sigui una jugada molt dolenta. En aquest joc una jugada molt dolenta suposa una pèrdua molt gran i determina molt el resultat. Per contra entre una bona jugada i una excel·lent la diferencia tendeix a ser molt poca. Aquest comportament implica que la cerca triga molt en arribar a convergir a una bona solució i, per tant, es preveu un èxit baix si s'hagués implementat aquest algorisme.

A més a més el joc no està preparat per fer simulacions de la situació del mapa, a molts llocs s'assumeixen moltes variables respecte l'estat actual del joc que intentar simular-les implicava haver de programar una estructura auxiliar molt gran.

Donat això es va valorar que l'algorisme no funcionaria prou bé i que a més el temps estimat era insuficient per a implementar-lo.

El segon problema trobat ha estat la manera en que s'enviaven les instruccions a les unitats. La primera versió de l'algorisme s'enviaven les instruccions a nivell batalló. La idea era que un batalló s'autogestionés per a complir els seus objectius però s'ha trobat els següents problemes.

S'han trobat molts casos en que les unitat, tot i estar juntes no tenen en la seva jugada bona perquè realitzar la mateixa acció. Es possible per exemple, voler retirar una unitat per ajudar a un altre batalló o perquè esta ferida, però amb la resta d'unitats seguir volent atacar.

Un segon problema que es va trobar va ser que la definició de batalló com a grup d'unitats implica que quan les unitats estan a una certa distància entre elles o bé no trobes cap batalló i llavors es tractarien individualment com s'està fent o totes les unitats estan a distància suficient entre elles com per considerar-les un batalló però quan compares les unitats dels extrems, aquestes estan tant lluny entre elles que la seva funció és totalment diferent. Això planteja un altre problema, el fet de que hi hagués una unitat o deixés d'estar feia que s'ajuntessin per exemple dos batallons que si no hagués estat, haguessin estat dos batallons diferents, tornant a trobar el mateix problema d'abans. El concepte de batalló és molt útil durant el procés d'anàlisis perquè si permet avaluar si es guanyarà el combat la majoria dels casos, o si el batalló

està més o menys ben col·locat amb cert encert però no permet una generalització del moviment de les unitats per batalló.

Finalment es conclou doncs que el concepte de batalló era mol útil alhora de decidir l'estat de la partida però no per controlar el moviment específic de cada unitat. Per això finalment l'estat de la partida s'analitza a partir dels batallons però cada unitat rep el conjunt d'instruccions a realitzar individualment.

7 Anàlisis i comparació

7.1 Comparació teòrica entre algorismes

Amb la descripció anterior es veuen clarament diferències entre les dues intel·ligències artificials.

La primera és la anàlisis de mapa, la intel·ligència artificial per defecte no fa cap tipus d'anàlisis de mapa i es comporta de la mateixa manera en qualsevol escenari mentre que la intel·ligència desenvolupada en aquest treball executa un anàlisis previ.

Una altre diferencia molt important és que l'ordre de les accions és absolut. Mentre que el meu heurístic calcula també l'ordre de les accions en l'altre intel·ligència l'ordre ve definit per la puntuació directament que té que es compleixi.

Mencionar també que la intel·ligència artificial per integrada en el joc té moltes possibles accions i incorpora molts més casos que la desenvolupada en el treball, és a dir, el seu combat és molt millor, igual amb detectar possibles amenaces cap a l'heroi. A més hi ha un petit detall que no s'ha mencionat abans i és que la intel·ligència per defecte es capaç d'executar prediccions quan més d'una unitat ataca a la mateixa unitat enemiga i calcular amb més exactitud la ponderació de l'atac que executarà una unitat individual. Tot i que està executant cooperativitat això només s'executa quan es valora si un combat és bo o dolent.

La intel·ligència per defecte no defineix els batallons ni agrupa les uniats alhora de valorar, simplement realitza la millor acció de cada unitat seguint l'algorisme proposat. En això també es diferència de la desenvolupada en aquest treball ja que, en aquesta, tenim una capa a un nivell superior que s'encarrega de valorar l'estat de la partida i donar instruccions individuals i especifiques a les unitats. Mentre que una permet aquesta flexibilitat alhora de decidir les instruccions per a les unitats l'altre només permet això de manera estàtica a la configuració.

Finalment, hi ha una diferència que té bastant impacte a la partida i és el valor que se li dona a les cases. L'heurístic desenvolupat dona molt més valor a les cases que al combat, molts pocs atacs s'executen per davant de capturar una casa enemiga. En canvi, si veiem l'ordre d'execució de les accions llistades a l'apartat anterior veiem com el combat està per davant de conquerir cases. Aquest canvi en les prioritats comporta execucions molts diferents entre les unitats dels diferents combatents.

7.2 Plantejament de la hipòtesis

Per observar els resultats obtinguts es plantegen les dues hipòtesis següents:

- Són equivalents? En aquest cas intentem determinar si les dues intel·ligències tenen un comportament equivalent a nivell de percentatge victòries, és a dir que els dos algorismes empaten.

- Juguen les dues intel·ligències al mateix nivell? En aquest cas intentem determinar si les dues intel·ligències juguen al mateix nivell. S'ha considerat que dos agents [5]juguen al mateix nivell si el seu percentatge de victòries envers l'altre està entre el 40% i el 60%. Si es supera aquests límits s'assumeix que un agent està jugant a un nivell superior que l'altre.

7.3 Metodologia

Per poder extreure els resultats s'ha desenvolupat un *script* que enfronta la intel·ligència artificial desenvolupada contra la del joc repetidament i enregistra el guanyador de la partida. Per fer-ho equitatiu es canvia el jugador que juga primer.

Aquest script s'executava cada cop que es volia saber si els nous canvis introduïts havien estat favorables o no. S'ha de tenir en compte que s'ha imposat la restricció de que i hagi només dos agents enfrontant-se entre sí.

7.4 Aleatorietat i complexitat

Un dels problemes més grans a l'hora de contrastar els resultats, per anar veient l'evolució en el programa, és la aleatorietat del resultat de l'execució i la complexitat del joc.

A nivell de complexitat, ens trobem amb moltes situacions que al preparar una millora pel programa, que es pensava que hauria de millorar els resultats, acaba empitjorant-los.

A més, es molt costos a nivell de temps saber si un canvi ha estat realment favorable o no, ja que s'han de fer moltes execucions del programa per poder veure la realitat dels canvis. Això es deu a la aleatorietat del joc i per poder treure conclusions vàlides s'ha estimat que menys de 100 partides no són significatives i fins les 500 partides no és comença a veure clarament si el canvi ha estat positiu o no.

[5] Agent intel·ligent: Entitat capaç de percebre un entorn, processar la informació rebuda i respondre. En el context del treball pot ser una de les dues intel·ligències artificials o un jugador humà.

7.5 Resultats

A continuació s'expliquen els diferents resultats que s'han anat obtenint durant el desenvolupament del treball, entre parèntesis el percentatge de victòries obtingut. Les versions s'expliquen de manera incremental de manera que s'entén que hereten els comportaments bons de la versió anterior.

- Primera versió (0%): Aquesta versió tenia implementat un algorisme molt diferent al final. Els camins es calculaven correctament, però el combat es calculava d'una manera molt més general, no es tenien en compte molts casos, les instruccions es feien a nivell de batalló i la funció que decidia i enviava era molt més complexa. El principal problema que presentava, era que el combat era massa dolent i la funció que enviava les instruccions intentava fer masses coses que acabaven sent molt difícil tractar-les a l'heurístic. A més l'algorisme que decidia els moviments a nivell unitat era força dolent.

- Segona versió (17%): Aquesta versió va suposar tot el replantejament d'estructura, on les instruccions s'enviaven a nivell d'unitat per simplificar el seu control. A més, aquesta versió, encara no tenia implementada la capa superior que decidia instruccions concretes. L'únic que feia era avançar per un camí escollit arbitràriament amb la majoria d'unitats, conquerir les cases i l'heurístic era capaç de valorar bé els combats. Tendia a perdre perquè l'heroi moria massa cops degut a que s'exposava massa o perquè cap unitat el defensava i quedava exposat ràpidament. Tot i així si aconseguia sobreviure suficient i la partida s'allargava aquesta tendia acabar guanyant-la ja que es feia suficient pressió per un dels camins.

- Tercera versió (40%): En aquesta versió, les bases del que seria l'algorisme final ja estaven implementades. L'heurístic valorava bé la majoria de situacions, però encara fallava en certs casos. Hi havia la arquitectura *shout-ahead* implementada, i s'havia millorat molt el comportament de l'heroi, per a evitar que l'heroi ataques igual que la resta d'unitats i es tirés davant les unitats enemigues quan tenia un bon atac. En aquest punt a més l'heurístic retornava puntuacions similars a les finals.

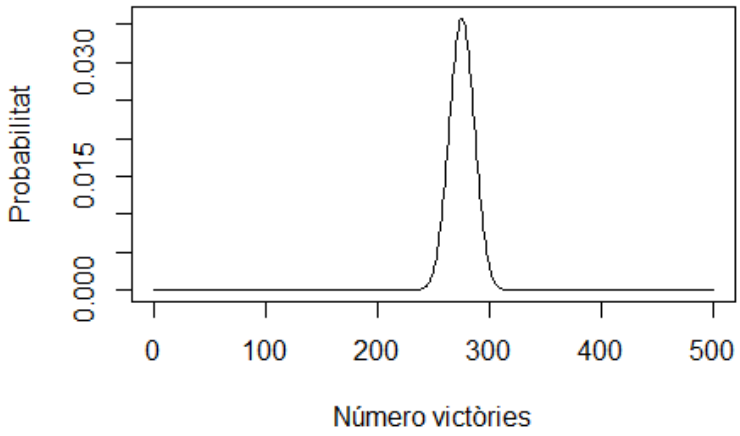
- Quarta versió (55%): Versió final de l'algorisme. En aquesta versió es va desestimar la arquitectura *shout-ahead* implementada, es van introduir diverses millores en el comportament i es van refinar molt les puntuacions de l'heurístic. És van fer petites millores en el combat per tenir en compte casos extrems que no s'havien contemplat abans. Es va millorar la interacció de les unitats amb les cases, de manera que les conquerien de manera més eficient, les protegeixen dels enemics i les fan servir per curar-se quan han rebut mal. En aquesta versió les unitats trien el camí correcte i

implementen l'estratègia explicada anteriorment. A més, les unitats són molt més conscients del risc al que s'exposen al avançar pel camí i són capaces de situar-se en lloc avantatjosos per defensar la posició quan escau.

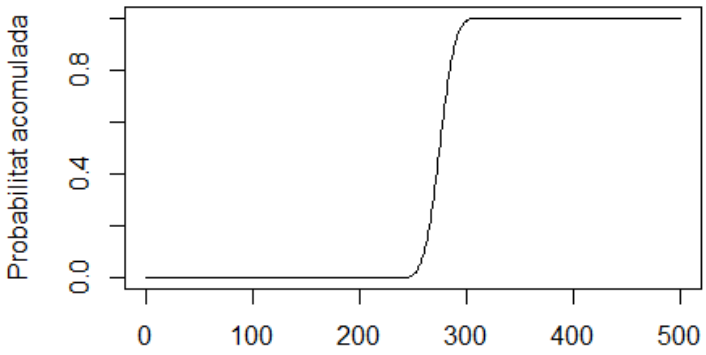
7.6 Anàlisis

Es fàcil veure que d'una execució només podem obtenir dos resultats que són si ha guanyat una intel·ligència o l'altre, per tant, tenim una distribució binomial.

Primer de tot es calcula la funció de distribució amb els paràmetres probabilitat de guanyar igual a 55% i amb un mostreig de mida 500 i obtenim la funció de probabilitat següent:



I la següent funció de distribució de probabilitat acumulada:



7 Funció de probabilitat acomulada

Si suposem que les dues intel·ligències són equivalents quina és la probabilitat d'obtenir el percentatge de victòries obtingut (*p-value*[6])?

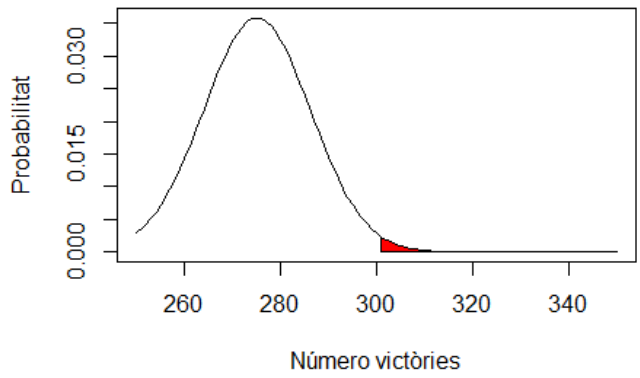
La probabilitat obtinguda és del 2.53%.

A continuació calculem l'interval de confiança del 95% per ambdós extrems:

L'interval normal de confiança del 95% en 500 execucions d'una distribució binomial amb probabilitat 0.55 és [0.506, 0.594]. Amb aquest interval es pot veure que la intel·ligència desenvolupada és lleugerament millor que la definida en el joc.

Calculem la probabilitat de que no juguin al mateix nivell:

En la següent gràfica es mostra visualment la probabilitat de que estiguin a nivells diferents on la intel·ligència desenvolupada en el treball guanyaria a la intel·ligència per defecte.



8 Funció de probabilitat (ampliada). En vermell la probabilitat de superar el 60% de victòries.

La probabilitat de tenir més d'un 60% de victòries havent obtingut un 55% en una mostra de 500 és la probabilitat d'obtenir més de 301 victòries. Calculem la probabilitat acumulada fins a aquest nombre de victòries i la restem a 1 per obtenir la probabilitat de que estiguin a nivells diferents, concretament a un nivell superior, ja que l' inferior és negligible.

La probabilitat acumulada fins al 301 és de 0.986, per tant la probabilitat de tenir realment una intel·ligència que jugui a un nivell superior és de 0.014, és a dir del 1,4%.

Amb això podem concloure que tot i no ser equivalents, no juguen a nivells molt diferents.

Per tant es rebutja la hipòtesis de que les dues siguin equivalents, el valor és molt petit i el 50% no entra en l'interval del 95% de confiança, i si s'accepta que juguin a un nivell semblant, ja que la probabilitat de que no ho facin és molt petita.

[6] P-value: probabilitat d'obtenir en un test d'una hipòtesis un resultat igual o més extrem que l'obtingut.

8 Desviaments en la planificació

8.1 Temps invertit per tasca

Finalment el temps invertit per desenvolupar cada tasca del treball ha estat el següent:

Tasca	Duració (setmanes)
GEP	6
Anàlisis del joc	2 (S)
Entendre el codi font del joc	1 (S) / 1
Buscar mètodes	1
Definir l'algorisme	2
Implementació	10
Observar i comparar els resultats	1
Extreure conclusions	1
Preparació de la defensa del projecte	1
Correcció	3
Redactar el document	1
Total	27

(S) = Setmanes solapades amb una altre tasca, no es sumen al total.

\- = Es realitza juntament amb la tasca realitzada.

8.2 Pla d'acció alternatiu.

Durant la implementació de l'algorisme han sorgit diversos errors que han fet que la durada d'aquest bloc s'allargués.

No s'ha aconseguit una intel·ligència artificial funcional per a la primera entrega perquè el temps empleat en descobrir com executar una intel·ligència artificial es va allargar una setmana més de l'esperat. Això va comportar el primer retràs en la planificació.

El segon problema que es va trobar va ser alhora de la implementació. Es va allargar una setmana més la implementació de l'algorisme degut a que la documentació no arriba a un nivell tant baix en les funcions. Per això es va dedicar una setmana, que no estava contemplada en la planificació, a entendre el codi. Per això per entregar el treball s'haurien necessitat dues setmanes més que a la planificació proposada.

Un cop en aquest punt, la planificació anterior contava que l'algorisme no tractés casos molt concrets i es centrés més en l'estratègia general. Aprofitant el temps extra per a

l'entrega s'ha millorat molt en el tractament dels casos específics que es donen en el joc donant com a resultat una notable millora en la intel·ligència artificial.

8.3 Descripció dels problemes trobats al implementar la intel·ligència artificial.

A continuació es descriuen detalladament els problemes que s'han trobat a nivell de projecte alhora de desenvolupar el treball.

El problema trobat alhora de modificar la intel·ligència artificial del joc per a poder jugar amb la IA desenvolupada en el treball és que el joc ja no està preparat ni s'ofereix documentació sobre com implementar una intel·ligència artificial amb c++. Això és degut a que es va veure que la majoria de desenvolupadors de mapes només volien canviar les regles que es feien servir en la IA per defecte, i per tant, es va desenvolupar un sistema amb *lua* que permet ràpidament modificar-les sense haver de compilar el joc de nou.

La documentació en c++ està desfasada i part del codi està en desús.

Per tant, per executar una intel·ligència artificial pròpia el que s'ha fet es buscar un mètode alternatiu.

La solució trobada ha estat la següent. Com que la intel·ligència artificial per defecte actual treballa amb *stages.* Aquests són executats per la IA per defecte i l'únic implementat actualment és el *main_loop stage.* Sabent això s'ha trobat la manera de substituir aquest *stage* per un creat per nosaltres. S'ha creat una classe que hereta de la classe *stage* i s'ha implementat la funció *play*_stage. És pot passar quina configuració es vol i amb això s'ha aconseguit que el joc faci servir la IA que nosaltres implementem encara que el nom de la IA sigui sent la de per defecte ja que estarà executant el nostre *stage*.

El problema de que la documentació no estava actualitzada s'ha resolt com s'ha comentat abans a dedicar més temps del programat a saber com extreure la informació necessària pel desenvolupament de la intel·ligència artificial. Es van mirar totes aquelles classes relacionades amb la intel·ligència artificial implementada per defecte per a poder ser capaç d'extreure la informació. Aquest temps també va servir per entendre millor quines elements del mapa són assumits com a correctes en certes funcions i quines actualitzacions comportava moure una unitat o fer un atac.

9 Cost final del projecte

El cost en recursos humans del projecte és:

- Programador: Finalment s'han fet 540h de projecte. He suposat un sou de 1.5 cops el preu mínim que estableix la universitat. 540 * 1.5 * 7.5 = 6075 €.

- Tutor: He suposat que el tutor li dedica 2h a la setmana de feina i que el seu sou per aquesta feina seria d'uns 20 €/h. 2 * 27 * 20 = 1080 €.

Per tant, en recursos humans es gastarien 7155€. Això són 2216 del pressupostat en recursos humans.

El cost en altres despeses és de:

- Connexió a internet: Es van pressupostar 90€ en internet i finalment s'ha gastat 27*5 = 135€.

- Electricitat: El cost final de l'electricitat s'ha estimat en 25*27/18 = 38€

Estimació del cost del projecte sense tenir en compte possibles incidències.

7155 + 100 + 135 + 38 = 7428 €

En l'estimació dels costos es menciona:

*Per això el pressupost real del projecte si el deslliguem de les entregues marcades per la facultat és de 5154 *2/18 = 5727.*

El projecte té un pressupost total de 5727 €.

El cost final del projecte ha superat en 1701€ el cost estimat d'aquest. Però s'ha de tenir en conte que el període d'implementació de l'algorisme s'ha allargat per millorar la qualitat del treball millorant la intel·ligència artificial degut a que l'entrega del projecte no es pot atraçar dues setmanes només.

9.1.1 Taula de costos reals

Font	Cost (€)
Programador	6075
Tutor	1080
Ordinadors	100
Connexió a internet	135
Electricitat	38
Total	7428

10 Integració de coneixements

El projecte està clarament relacionat, com es obvi, amb la intel·ligència artificial. Es tracten i s'analitzen diferents algorismes d'intel·ligència artificial. S'han tractat en més o menys profunditat tots els algorismes exposats a l'inici del treball i s'ha avaluat la seva utilitat alhora d'implementar la intel·ligència artificial per aquest joc.

Durant la implementació s'han tocat temes d'estructures de dades, per a fer eficient les funcions moltes implementen una cache que en cas d'haver-se calculat abans el resultat el serveixen directament sense tornar-lo a calcular. Això és essencial per qualsevol problema però encara més pels videojocs on es disposa molt poc temps d'execució.

També s'ha fet servir per l'anàlisi del mapa tècniques de geometria computacional per tal d'analitzar la posició relativa de les tropes respecte les bases.

En la anàlisis dels resultats també s'ha fet servir teoria de probabilitat i estadística.

Per últim en aquest treball s'integrarà codi propi a un projecte real molt més gran, per tant, s'ha d'entendre com funciona aquest per ser capaç de modificar-lo. He observat la importància d'una bona documentació, organització i de tenir una bona estructura tant de l'arbre del projecte com de codi.

10.1 Tecnologies

El treball s'ha desenvolupat amb c++ ja que el codi de *battle for wesnoth* està amb c++. Hi havia altres alternatives com *lua* o *python* ja que *battle for wesnoth* té mòduls d'integració però no oferien totes les possibilitats que té c++. Aquests mòduls donen als creadors de mapes una manera de crear la seva intel·ligència artificial amb certes funcions que proporcionen però el nivell de tractament que s'ha necessitat per aquest projecte fa impossible implementar la intel·ligència amb aquests mòduls.

Per fer el petit programa que executava una sèrie de partides s'ha fer servir el llenguatge *bash*.

També s'ha fet servir R per a generar les gràfiques i el càlcul de resultats.

11 Conclusions

Durant el desenvolupament del treball s'han intentat diverses maneres de desenvolupar la intel·ligència artificial. S'ha vist que és realment *Battle for Wesnoth* suposa un repte computacional important des de el punt de vista de la intel·ligència artificial. La diversitat del joc és enorme i proposa molts casos que suposen tot un repte alhora d'intentar generalitzar i d'abstreure els punts claus del joc.

El percentatge de victòries ha estat molt satisfactori ja que en un inici no s'esperava superar la intel·ligència artificial del joc. S'ha vist que, tot i lluitar pitjor que la intel·ligència base, a la llarga l'estratègia desenvolupada millora el comportament de la intel·ligència artificial. Tot i que en cap moment s'ha pensat específicament en contrarestar l'estratègia que desenvolupa la intel·ligència artificial per defecte, s'ha de ser conscient que al haver estat entrenant la intel·ligència artificial desenvolupada amb partides on jugava contra la defecte, és possible que hagi quedat sobre ajustada i el seu rendiment sigui pitjor contra altres. Per això, seria interessant ara realitzar partides contra altres intel·ligències o persones humanes per veure el seu rendiment.

Aquest treball m'ha servit per plantejar-me el repte de la cooperativitat entre unitats i entendre la dificultat que suposa coordinar diversos agents. També la dificultat que hi ha alhora de desenvolupar un heurístic complex. Com a exploració del camp de la intel·ligència artificial considero que ha estat un treball encertat i donat que es va plantejar amb aquesta idea d'exploració personal el considero també tot un succés en aquest sentit.

Com a conclusió final dir que tot i que la intel·ligència artificial desenvolupada està igualada amb la intel·ligència artificial donada en el joc, encara hi ha molt marge a millorar amb moltes possibles extensions, casos concrets que es podrien solucionar d'una manera molt més acurada i provar moltes altres opcions que hi ha alhora de desenvolupar l'algorisme, tant a nivell d'estratègia com a nivell d'implementació.

12 Referències

- Pàgina oficial del joc:

[1] https://www.wesnoth.org/

- Pàgina oficial del joc per a desenvolupadors:

[2] https://wiki.wesnoth.org/DeveloperGuide

- Pàgina oficial del joc per a jugadors on s'expliquen algunes de les tàctiques a tenir en conte:

[3] https://wiki.wesnoth.org/AdvancedTactics

- Pàgina que conté enllaç al llibre on s'explica el context actual de la intel·ligència artificial aplicada a jocs:

[4] http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=4804728

- Conferència realitzada sobre intel·ligència artificial:

[5] http://www.ieee-cig.org/

- Informació de Montecarlo Tree Search.

[6] https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/

Autor: https://www.caktusgroup.com/about/jeff-bradberry/

- Resultats obtinguts d'implementar la IA amb arquitectura *shout-ahead*.

[7] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6633608&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6613622%2F6633607%2F06633608.pdf%3Farnumber%3D6633608

- Informació sobre la cerca heurística

[8] http://www.cs.upc.edu/~bejar/ia/transpas/teoria/2-BH2-Busqueda_heuristica.pdf

-Informació sobre la intel·ligència artificial aplicada a jocs

[9] http://www.cs.upc.edu/~bejar/ia/transpas/teoria/2-BH4-juegos.pdf

Autor: Javier Béjar http://directori.upc.edu/directori/dadesPersona.jsp?id=1001970

- Càlcul de l'àrea orientada

[10]

http://dccg.upc.edu/people/vera/wp-content/uploads/2013/06/GeoC-OrientationTests.pdf

Autor: Vera Sacristián
http://directori.upc.edu/directori/dadesPersona.jsp?id=1000525

[11]

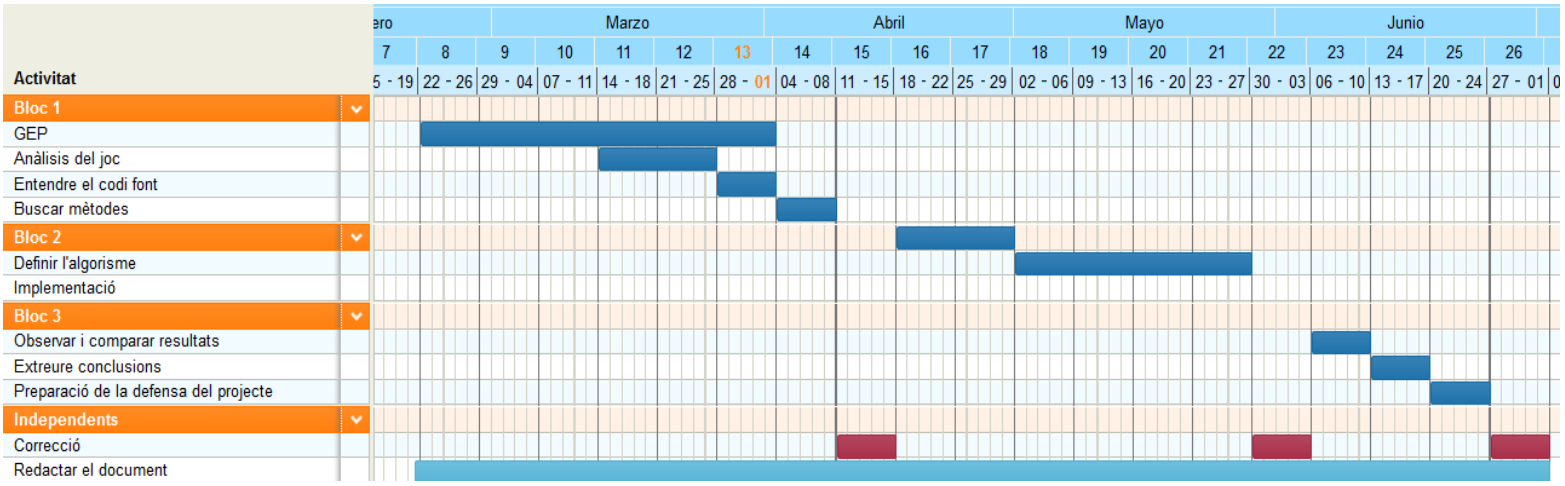
-Probabilitat i estadística, binomial

http://www-eio.upc.es/teaching/pe/formulari_B456.pdf

Autor: No definit. Pàgina oficial de probabilitat i estadística de la facultat d'informàtica de la UPC.

Annex

Diagrama de Gantt (Planificació inicial).

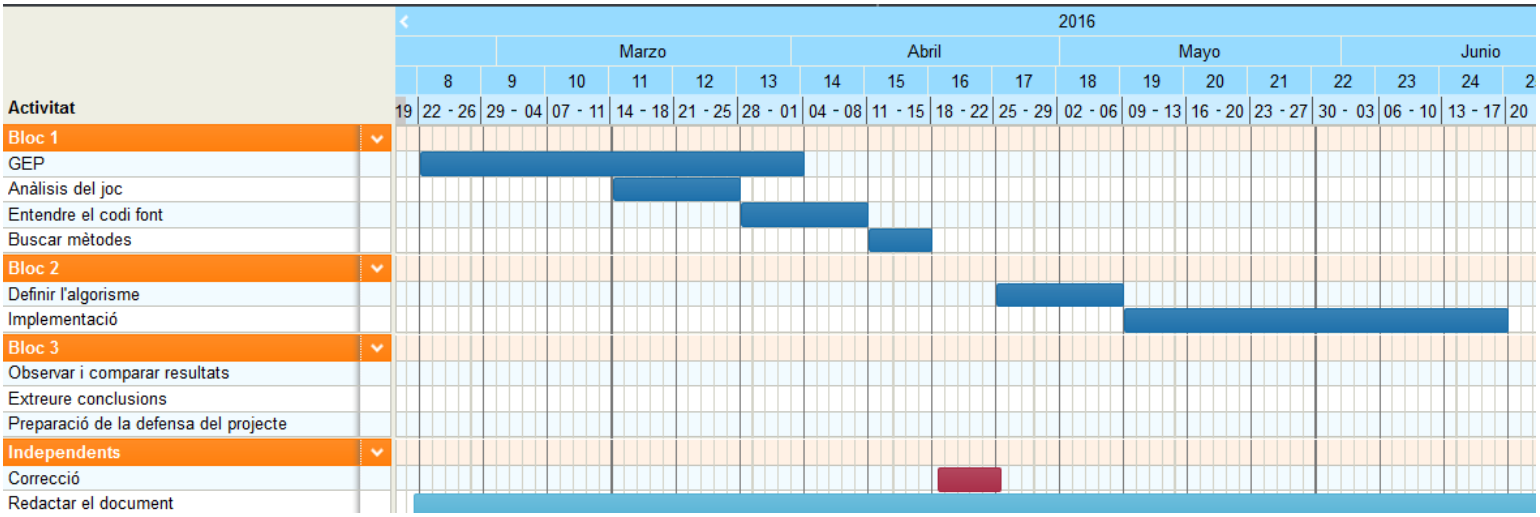


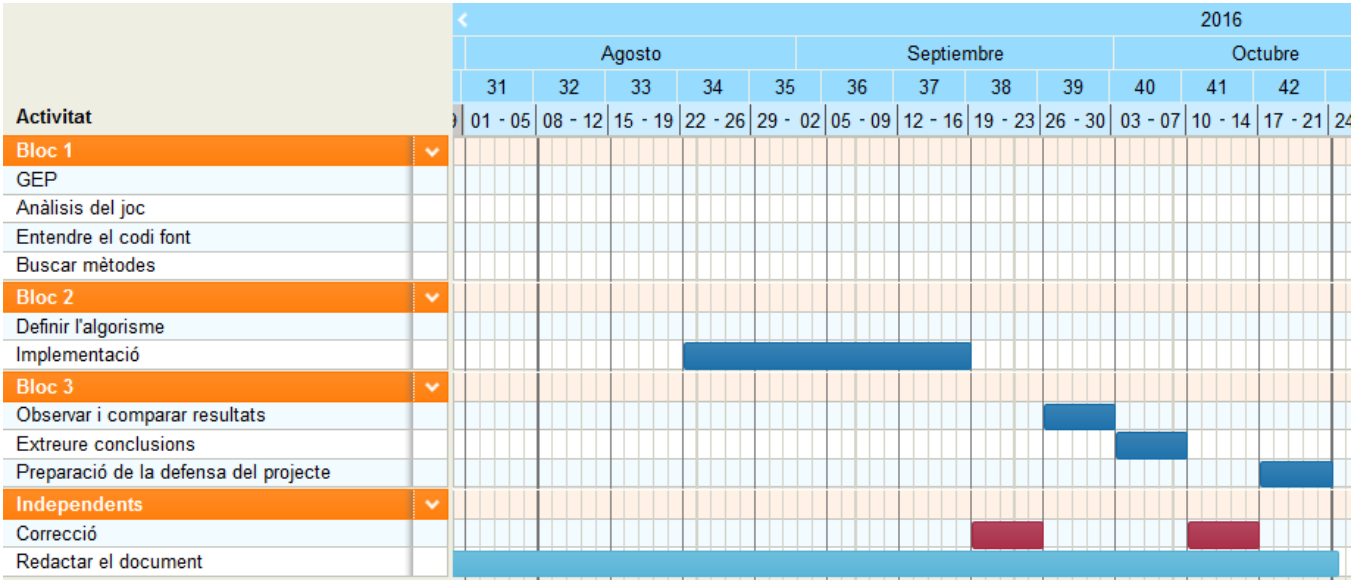
Cada dia representa 4h de feina.

La setmana 8 és la setmana que comença el 22 de Febrer.

Acaba just abans de la setmana del 27 de Juny, setmana on hi ha les presentacions del treball.

Diagrama de Grant (Planificació final)





Cada dia representa 4h de feina.

La setmana 8 és la setmana que comença el 22 de Febrer.

Acaba just abans de la setmana del 21 d'Octubre.

*els dies que no es veuen al diagrama estan buits.